CONFIGURATION OF MULTI-CLUSTER PROCESSOR FROM SINGLE WIDE THREAD TO TWO HALF-WIDTH THREADS

BACKGROUND OF THE INVENTION

[0001] Certain VLSI processor architectures now group execution units as clusters to process bundled instructions. One "bundle" of instructions has three instructions; a cluster operates to process one bundle, or more, of instructions. FIG. 1 illustrates the prior art by showing a processor register file 10 coupled with clusters 12(1), 12(2)... 12(N). Each cluster 12 is a physical logic unit that includes multiple pipelines, with bypassing (e.g., as shown in FIG. 2), to parallel process the multiple instructions within the bundles. The advantages of clusters 12 lie primarily in timing efficiencies. Each cluster 12 more quickly processes one bundle as compared to two bundles; appropriate use of clusters may reduce bypassing requirements within processor architectures. However, a loss of performance is also realized in cluster-based architectures when information is shared between clusters, as at least one cycle latency results from moving data between them.

[0002] Certain VLSI processor architectures also use "multi-threading" techniques to process instructions through pipeline stages. FIG. 2 shows one exemplary multi-threading architecture 20 of the prior art. Architecture 20 illustratively has two program counters 22(1), 22(2), an instruction fetch unit 24, a multiplexer 26, a plurality of pipelines 28(1), 28(2)...28(N), bypass logic 30, and register file 10. Multiple program counters 22 provide for the multiple program "threads" through pipelines 28; as any one instruction stalls, another instruction may proceed through pipelines 28 to increase collective instruction throughput. As known in the art, each counter 22 is a register that is written with the address of the next instruction at the end of each instruction fetch cycle in the pipeline; each pipeline 28 includes multiple execution stages such as fetch stage F, the decode stage D, the execute stage E, and the write-back stage W. Individual stages of pipelines 28 may transfer speculative data to other execution units through bypass logic 30 and multiplexer 26 to reduce data hazards in providing data forwarding capability for

architecture 20. Register file 10 is typically written to, or "loaded," at the write-back stage W on logic lines 32.

[0003] The invention advances the state of the art in processing architectures incorporating logic such as shown in FIG. 1 and FIG. 2 by providing methods and systems for processing multi-thread instructions through clustered execution units. Several other features of the invention are apparent within the description that follows.

SUMMARY OF THE INVENTION

[0004] The invention of one aspect processes bundles of instructions preferentially through clusters such that bypassing is substantially maintained within a single cluster. Alternatively, in another aspect, the invention processes bundles of instructions preferentially through multiple clusters, with bypassing therebetween, to increase "per thread" performance. The cluster architectures of the invention thus preferably include capability to process "multi-threaded" instructions.

[0005] In one preferred aspect, the invention provides a "configurable" processor architecture that operates in one of two modes: in a "wide" mode of operation, the processor's internal clusters collectively process bundled instructions of one thread of a program at the same time; in a "throughput" mode of operation, those clusters independently process instruction bundles of separate program threads. Accordingly, the invention of this aspect provides advantages by flexibly operating with (a) a high degree of parallelism (i.e., in the "throughput" mode) or alternatively (b) a high degree of single threaded performance (i.e., in the "wide" mode). An independent user desiring maximum single thread performance can therefore select the wide mode preferentially; another user desiring to process many orders simultaneously, and in real-time (e.g., in a business such as an airline company), can therefore select the throughput mode preferentially.

[0006] The invention is next described further in connection with preferred embodiments, and it will become apparent that various additions, subtractions, and modifications can be made by those skilled in the art without departing from the scope of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] A more complete understanding of the invention may be obtained by reference to the drawings, in which:

[0008] FIG. 1 illustrates cluster and register file logic for a prior art processor architecture;

[0009] FIG. 2 illustrates pipeline stage processing of multiply-threaded instructions relative to the register file of FIG. 1;

[0010] FIG. 3 schematically shows a multi-threaded cluster processing architecture of the invention;

[0011] FIG. 4 shows a flowchart for processing bundled instructions through a CPU in accord with the invention; and

[0012] FIG. 5 shows a schematic diagram of one cluster architecture of the invention.

DETAILED DESCRIPTION OF THE DRAWINGS

[0013] FIG. 3 shows a processor architecture 100 of the invention. Architecture 100 utilizes multiple clusters 102 and program counters 104 to process instruction bundles relative to a singly- or multiply-threaded program control. Each cluster 102 is illustratively shown with multiple pipelines 103(1)-103(n), which operate to process bundled instructions. A processor 106 connects to a memory 108 by a system bus 110 that carries data and control signals between processor 106 and memory 108. Memory 108 may for example store instructions to be performed by processor 106. Instruction cache 112 stores these instructions, and other processor-generated instructions, for processing within processor 106. In the preferred embodiment of the invention, architecture 100 is configurable to operate in a wide mode of operation; alternatively, architecture 100 operates in a throughput mode of operation. Processor 106 is configured to operate in either mode by a user-selected configuration bit, for example communicated to processor 106 through a configuration command 118. Processor 106 has at least one register file 120; however, architecture 100 may include multiple register files 120(1)-120(N) to operate with each of clusters 102(1)-102(N) as a matter of design choice.

[0014] Program instructions are decoded in the thread decode unit 130. Depending on the configuration bit, decode unit 130 detects and then distributes bundled instructions to program counters 104 according to the threads associated with the instructions. If the configuration bit is set to wide mode, then bundled instructions from the same thread are processed through multiple clusters 102 at the same time. If the configuration bit is set to throughput mode, then bundled instructions from one thread are processed through one program counter 104, and through a corresponding cluster 102; bundled instructions from other threads are likewise processed through another program counter and cluster pair 104, 102. An instruction memory 132 may optionally function to store bundled instructions, or to multiplex bundled instructions by and between different program counters 104 and different clusters 102, as a matter of design choice.

[0015] By way of example, in the throughput mode, three instructions from a single thread are bundled, by thread decode unit 130, and then processed through program counter and cluster 104(1), 102(1); three instructions from another thread are bundled, by thread decode unit 130, and processed through program counter and cluster 104(2), 102(2).

[0016] Each cluster 102 includes several pipelines and stage execution units so as to simultaneously perform, for example, F,D,E,W on multiple instructions within the bundle.

[0017] FIG. 4 shows a flow chart 200 illustrating functional steps in processing bundled instructions through cluster architectures on one embodiment of the invention. At step 202, a configuration bit is set to command the CPU to process bundled instructions within the wide mode or throughput mode. Generally, step 202 is performed at the beginning of operations, according to user commands of a computing system. Alternatively, a cluster within the CPU may select the configuration bit depending upon the occurrence or non-occurrence of certain criteria. Instructions are cached at step 203; for example, at step 203, instructions may be fetched according to thread association. A branch occurs at step 204 depending on the configuration bit. In the wide mode, at step 206, instructions of a common thread are bundled; these bundled instructions are then processed through multiple clusters, at step 208, to enhance "per thread" processing performance. In the throughput mode, at

step 210, instructions of a common thread are bundled; these bundled instructions are then processed through a single cluster, at step 212, to enhance multiple program instruction throughput through the CPU. After steps 208 or 212, the process is repeated at instruction fetch step 203, as shown, to process additional bundles through the CPU.

[0018] FIG. 5 illustrates a four-bundle cluster processor architecture 300 suitable for use with the invention. Architecture 300 includes two processing cores 302A and 302B. Each core 302 includes an associated register file 304 and pipeline execution units 306, as shown. Execution units 306 include internal bypassing capability, as indicated by arrows 308A, 308B. Cores 302A, 302B may be identical. Units 306A, 306B write-back to register file 304 by control lines 310A, 310B, respectively. If required, data transfer between cores 302 may occur via multiplexers 312A, 312B, as shown; a latch 314 may be used to couple data from one core 302 to the execution units 306 of the other core. Data from one core 302 that is architected to the register file 204 of the other core may be written, as shown, through a latch 316.

[0019] Each core 302 functions as a cluster, in accordance with the invention. In the wide mode, one thread may for example execute four bundles through both cores 302; inter-cluster communication occurs, with cycle delays, through multiplexers 312. In the wide mode, for example, core 302A may execute instructions corresponding to even program counter steps 0,2,4, etc.; and core 302B may execute instructions corresponding to odd program counters steps 1,3,5, etc. The cycle delays are eliminated through multiplexers 312 when architecture 300 operates in the throughput mode, as instruction bundles of common threads are only executed on a single core 302. The following illustrate how four bundles may be processed through architecture 300:

(1) 1 thread, 4 bundles, 2 clusters

(2) 1 thread, 2 bundles, 1 cluster; another thread, 2 bundles, other cluster

(3) 2 threads, 2 bundles, 1 cluster; 2 threads, 2 bundles, other cluster.

Note that (1) offers maximum parallelism in processing instructions, (3) offers maximum throughput of four separate threads, and (2) offers a mix between (1) and (3).

[0020] The invention thus attains the features set forth above, among those apparent from the preceding description. Since certain changes may be made in the

above methods and systems without departing from the scope of the invention, it is intended that all matter contained in the above description or shown in the accompanying drawing be interpreted as illustrative and not in a limiting sense. It is also to be understood that the following claims are to cover all generic and specific features of the invention described herein, and all statements of the scope of the invention which, as a matter of language, might be said to fall there between.